La programación gráfica a diferencia de los ejemplos que podemos ver para los clásicos ejercicios de “hacer una calculadora” o completar una ecuación de segundo grado, tiene que ver con la forma en cómo pintamos las cosas.

Aunque en esencia es igual en términos de los comandos y la manera de codificar, el paradigma y la filosofía cambia trascendentalmente. A diferencia de la primera en donde el objetivo es la presentación de comandos uno después del otro, y donde lo importante es el resultado final; la programación gráfica trata de “pintar” por “frames” , en donde cada vez que se necesite cambiar algo, se deberá borrar el frame anterior y colocar el nuevo.

Normalmente los entornos de programación que nos permiten generar esto ya vienen optimizados para programar de manera gráfica con “funciones” que ya dan oportunidad de pensar con respecto a cuadros en lugar de simples comandos secuenciados.

Para empezar a ver programacion y eventualmente los temas asociados como matemáticas discretas, matemáticas, arquitectura computacional, etc. vamos a utilizar la herramienta “Processing”, descargable desde: https://processing.org/download/

Cuando se baje se mostrará la siguiente pantalla.

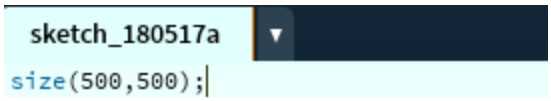
La parte negra es “la consola” o el lugar donde nos podremos enterar de lo que ocurre en el backstage. El espacio blanco es donde introduciremos el código.

Advertencia: muchas de las cosas al inicio son particularmente “Fe” , es decir, de momento hay que considerarlas ciertas, hasta que podamos tener conocimientos suficientes para entenderlas a fondo.

Ejercicio 1:

Se generará una pantalla en donde poder pintar. En este pedazo de código se explicará el concepto de función de manera muy ligera y eventualmente el de variable.

Para hacer más fácil el acercamiento, se empezará con un ejercicio que “parece” una simple secuencia de comandos y lo modificaremos hacia el pensamiento por frames.



Colocando esa línea de código y dando click en “PLAY” en la parte superior, se deberá colocar una pantalla sobre todo el código de color gris.

Para cambiar el color de esa pantalla, se puede utilizar la “función” background.

Aparecerá la misma ventana pero de color rojo oscuro.

Concepto de función: una función es una “maquinita” que recibe entradas y genera salidas. Las entradas están denotadas entre los paréntesis. De momento, todas serán números, aunque hay de dos tipos: números y “conjuntos de letras” (el espacio también es una letra). Las salidas pueden ser de dos tipos, implícitas y explícitas. Las implícitas es porque la función “hace algo” internamente. Por ejemplo la función background cambia el color de la pantalla. Las explícitas “escupirán” un valor desde el código para usarlas para otra cosa, como ejemplo está la función cos(X) que genera el coseno del ángulo enviado entre paréntesis.

Así como está la función background o size, existe la función rect que tiene la siguiente “firma”, es decir, la manera en cómo se usa, desde la referencia, es así:

rect(x , y , width ,height).

Lo cual significa que al usarla, los primeros dos números significa en donde se colocará la esquina superior izquierda y los segundos dos, es que tanto se extenderá hacia la derecha y hacia abajo (respectivamente).

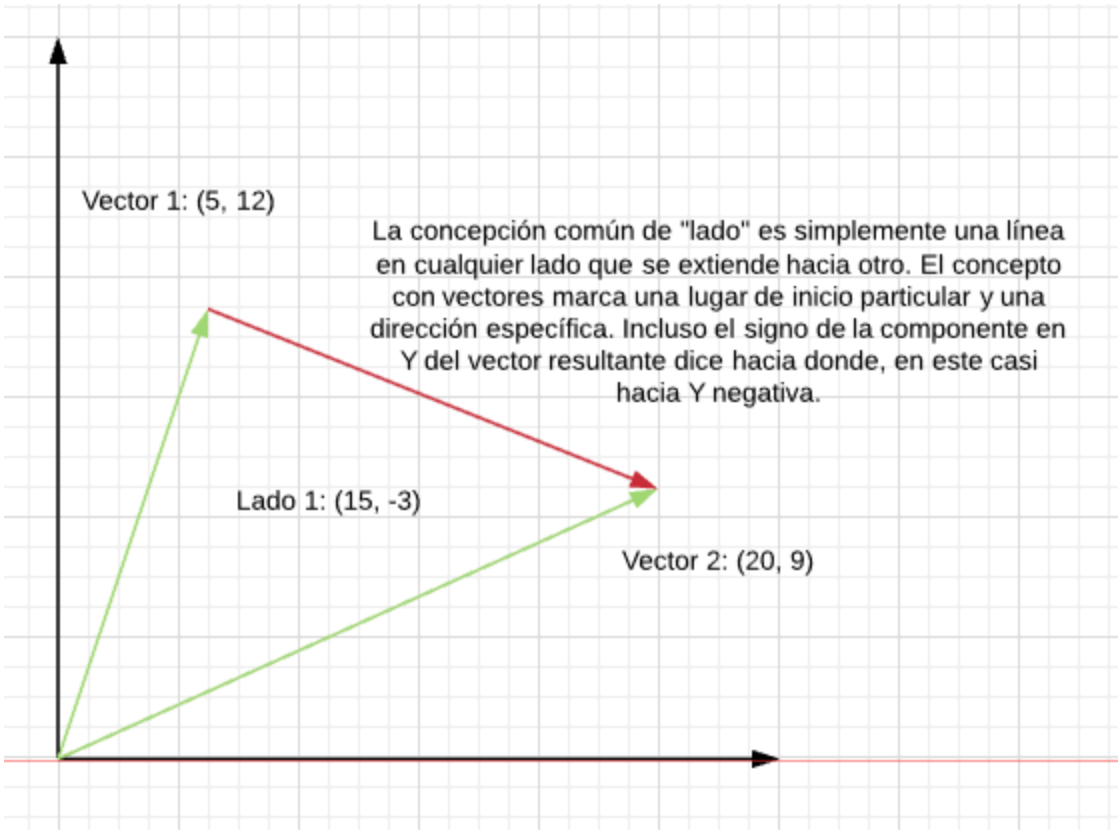
Inténtalo:
rect(40,40,100,100);

Reto: intenta hacer una cara con rectángulos y la función *ellipse* cuya firma es:

ellipse(x,y, width, height);

Nota de matemáticas:

Aunque desde la primaria se ha intentado establecer que una figura geométrica está compuesta de líneas, esta definición no es suficiente para 3D. De esta manera, a partir de esta sección, se podrán expresar las componentes de una figura geométrica como “Vectores”. Es decir, elementos con una dirección y magnitud. En realidad, se necesitan dos vectores para hacer una línea. Los dos vectores se entenderán como dos puntos en el espacio y la resta de ellos, generará el tercer Vector que es la línea que nos interesa.



Es muy importante analizar que el entendimiento que se tenía de un “LADO” se amplía hacia la generación del mismo via Vectores. El Vector resultante tiene varias características deseables, es claro si va hacia arriba o hacia abajo en Y, si va hacia la izquierda o la derecha, e incluso tiene el concepto de “NORMAL”, que es una línea perpendicular. Esta normal es la pendiente inversa. Particularmente para choques de esferas en 2D, este cálculo es muy fácil y genera el efecto que se necesita.

—

Toda vez que las funciones sirven para indicarle al “Sistema” (por sistema entiéndase sistema operativo, o un game engine, etc) que debe realizar una operación, las funciones también tienen un formato en donde nos “sirven” para calcular un resultado y posteriormente utilizarlo.

Dentro de la programación el concepto que es básico junto con el de las funciones, es el de variable. En realidad las variables son valores, valores a los que el programador “bautiza” para que en lugar de usar un número o un texto, se refiera a ellos como un elemento de “lenguaje común” con lo que la programación se transforma hacia un formato de comunicación entre la computadora y el usuario programador.

Definición de constante: los valores fijos como números o textos, se llaman constantes en la programación. Los siguientes son ejemplos de constantes: 1,2,556.425,1234465687,”3242”,”hola!”. Nótese que los últimos dos están entre comillas por lo que su “valor”, no es un número, sino es una palabra. La diferencia entre los textos y los números se ilustra en la siguiente tabla:

Combinación	+	-	*	/
Número con Número	suma	resta	multiplica	divide
Número con Texto	concatena	No aplica	No aplica	No aplica
Texto con texto	concatena	No aplica	No aplica	No aplica

Ejemplo

Combinación	+	-	*	/
1,5	6	-4	5	.2 (solo si uno de ellos es decimal, mas adelante se detalla)
345, “2345”	“3452345”	No aplica	No aplica	No aplica
“1323”,”jolgs”	“1323jolgs”	No aplica	No aplica	No aplica

Notar de manera importante que un número sumado con una cadena de texto, se convierte en texto.

Definición de variable: un valor al que se le coloca un “alias”. Este alias puede variar su valor siempre y cuando lo asignemos a otra constante, perdiendo la primera constante. Es común asociar el significado de variable computacional con variable matemática, sin embargo, una computadora no puede calcular ningún resultado si una variable no tiene un valor. Por su cuenta las matemáticas pueden determinar fórmulas por que contemplan el concepto de “el conjunto de todos los valores posibles”.

Las computadoras, sin las instrucciones adecuadas son inútiles. Entre más específicos sean los comandos, mejor funcionarán. De esta manera, la computadora dentro de muchos lenguajes de programaciçon ha subdividido los valores de las constantes en los siguientes:

Valor	Nombre	Descripción
1	int	Entero cuyo máximo valor es una conversión binaria del máximo en 4 bytes (detallado más adelante)
8.6	float	Decimal cuyo máximo valor es un número en formato decimal en 4 bytes
1235678908763213435789	long	Entero cuyo máximo valor es la conversión binaria de 8 bytes.
123456.32456789087324132489	double	Decimal cuyo máximo valor es un número en formato decimal en 8 bytes
“Texto fdsf fdsfa ”	String	Cadena de texto que puede ser desde una letra o espacio, hasta todo un libro, o incluso código de programación (en formato texto).
‘c’	char	Dato en 2 bytes que establece la conversión de un número a una letra. El ‘1’ como texto es distinto en valor al 1 como número.
Existen otros valores como short, byte, etc. Los anteriores son los más comunes y son los que se estarán utilizando.		

Ver: https://www.tutorialspoint.com/java/java_basic_datatypes.htm

Pregunta: ¡¿qué pasa si se suman, restan o multiplican los tipos de datos anteriores?

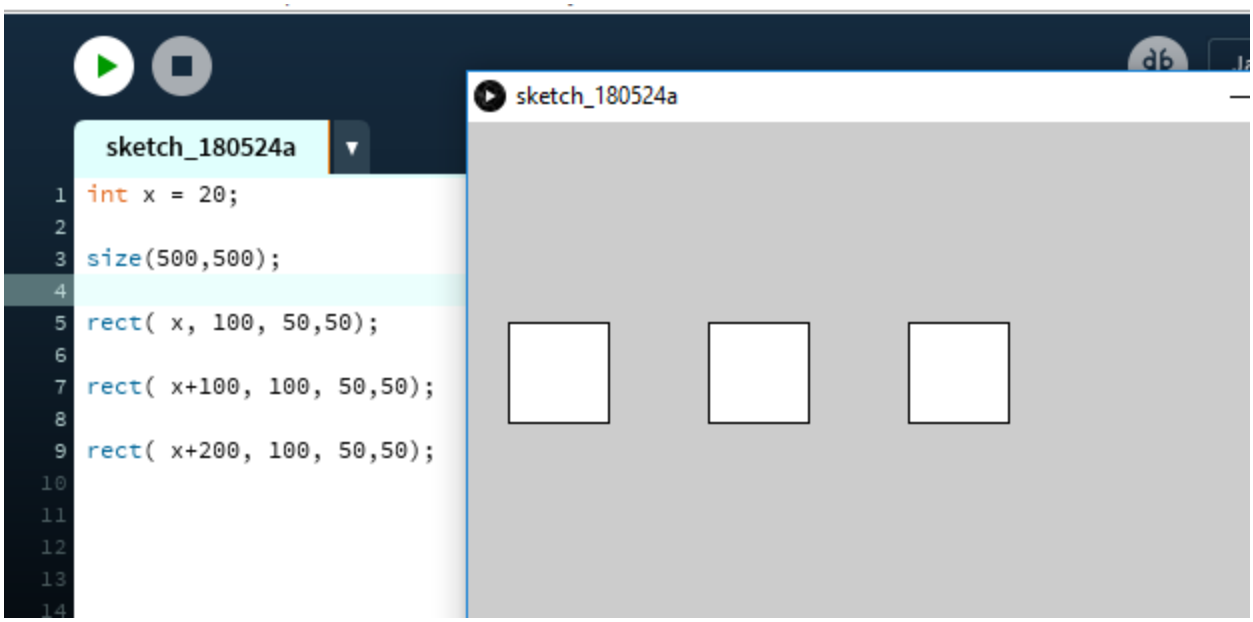
Las variables, particularmente las enteras sirven para muchas cosas, dentro de ellas están:

- La posibilidad de basar toda una programación de comandos en ellas, y fácilmente cambiar el valor inicial
- La abstracción de nombres comune al programador en lugar de valores que para un segundo programador, no son claros
- La abstracción de significar “cosas” que en la vida real dan pauta a la creación de abstracciones más complejas.

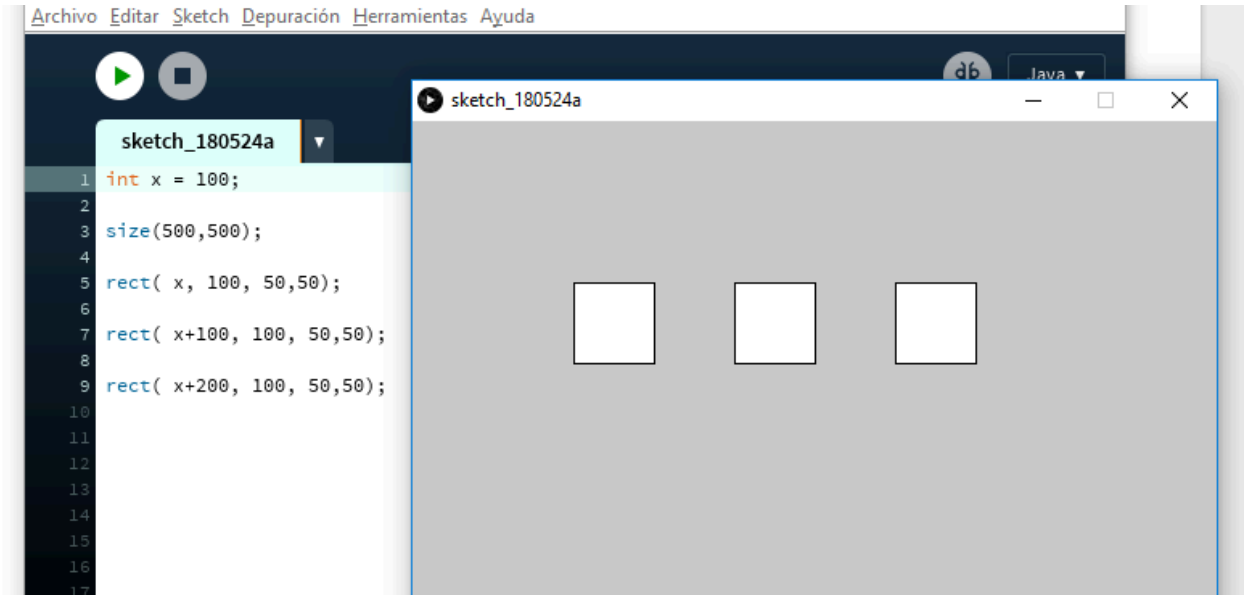
Intenta esto en Processing:

Como se puede ver, se ha sustituido el valor directo de la constante, por una variable, que en el orden arriba-abajo, adquiere un valor y luego ese valor se sustituye en la función.

Ejercicio: cambiar la cara que se hizo en el ejercicio pasado para que dependa de una variable y que cuando se quiera pintar en otro lado, solo se cambie el valor de esa variable. Por ejemplo:



Que luego fácilmente puede cambiar a:



Se nota la diferencia entre la distancia del primer cuadro al borde inicial de la ventana.

Ejercicio pro: modifica y genera otra variable para Y. Que se puedan cambiar ambas para poder “mover” la cara de un punto en el espacio a otro.

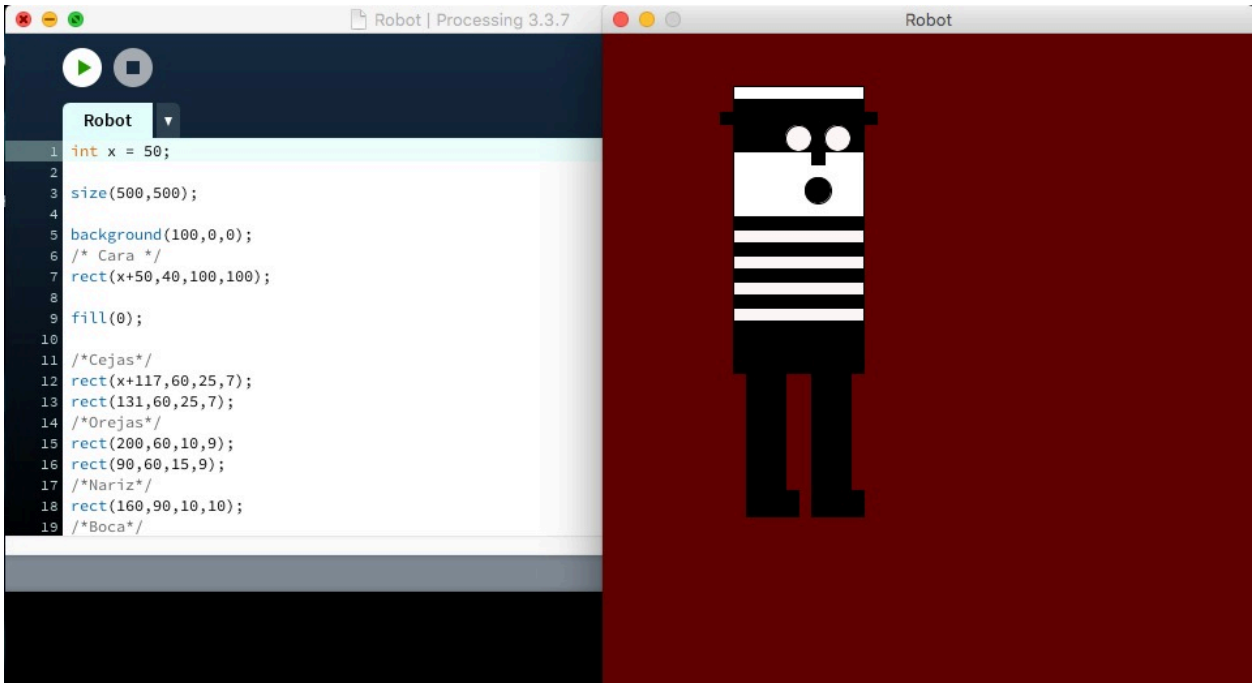
Es por más interesante que se pueden combinar las funciones con las variables para obtener ideas interesantes. Intenta:

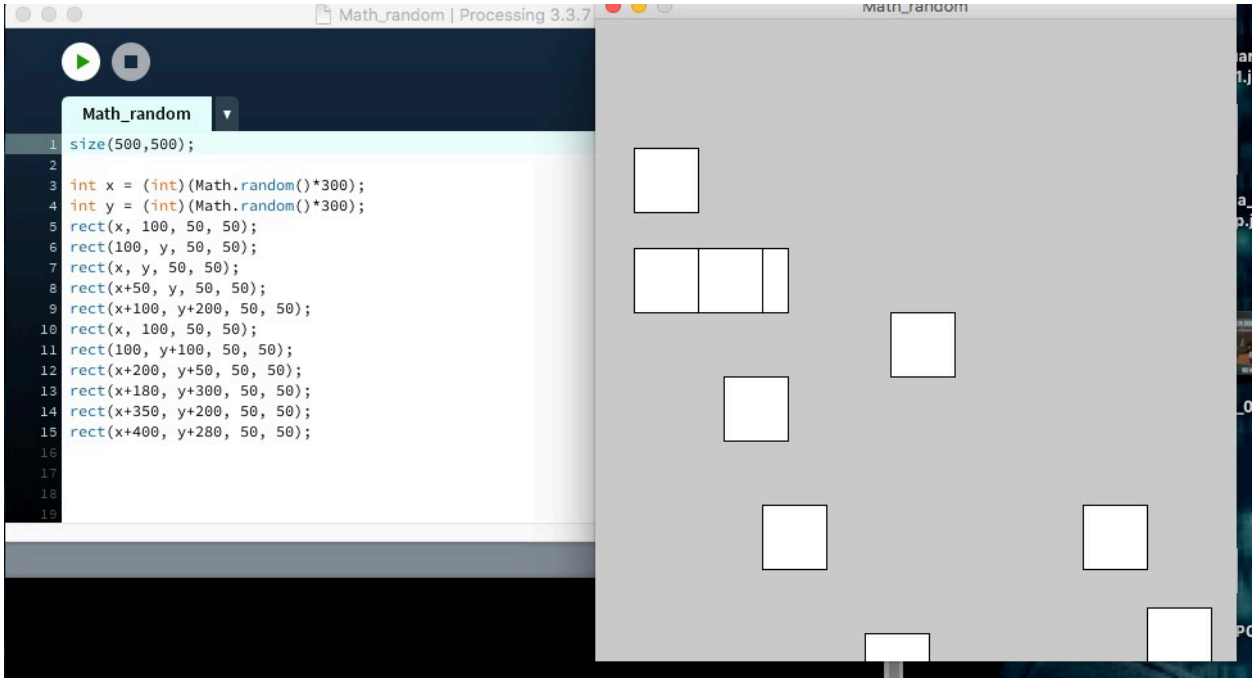
```
size(500,500);

int x = (int)(Math.random()*500);

rect(x, 100, 50,50);
```

En el código anterior, primero se ejecuta una función para establecer el tamaño de la ventana.

Después, se utiliza una función para obtener un número aleatorio. El número aleatorio es un valor decimal double. El problema es que no se pueden tener “pixeles decimales” entonces habrá que convertir el valor a “int”. Como el valor aleatorio puede ser algo entre 0 y 1, se multiplica por 500 para que el resultado sea un valor entre 0 y 500. Esto es muy usado en videojuegos para colocar enemigos en un mapa o tener items de valor distinto. Una vez que el valor ha sido conseguido, se mantiene como double. Para “cambiarlo” a “int”, se utiliza el mecanismo mostrado “(int)” conocido como “casting”. Es un cambio que el programador hace “a fuerza” y que el sistema respeta. Con ese valor, se puede asignar un “int” y luego usarlo para pintar el rectángulo.

Ejercicio: genera 10 cuadros que de manera aleatoria se coloquen en toda la pantalla.
Se deberán colocar tanto en horizontal como en vertical, aleatoriamente,
Extra: investigar la manera de cambiar el “relleno” de los cuadros, y hacer que cada cuadro tenga un color distinto.





Ejercicio resuelto.

ANEXO: código para cuadros con rellenos aleatorios:

```
size(500,500);
int px = 10;
int py = 10;
int r = 0;
int g = 0;
int b = 0;
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
```

```
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
px = (int)(Math.random()*width);
py = (int)(Math.random()*height);
r = (int)(Math.random()*255);
g = (int)(Math.random()*255);
b = (int)(Math.random()*255);
fill( r , g, b);
rect(px, py, 50,50);
```

–

Nota matemática

Sistema binario y sistema decimal

El sistema binario es una manera de ver el mundo en donde en lugar de haber dígitos del 0 al 9 (10 dígitos) van del 0 al 1 (sólo 2 dígitos).

En una computadora este sistema funciona puesto que la memoria RAM y los “registros” de los procesadores (mini memoria en el procesador) son literalmente espacios que pueden o no tener carga. Esto se puede implementar con capacitores o con circuitería donde una carga se mantenga fluyendo. Conceptualmente, esto se traduce como un 1 (carga) y 0 (no carga). Aunque los detalles electrónicos no son importantes para hacer operaciones binarias, sí es importante que se entienda la base en la computación del sistema binario.

Contando con este elemento y sabiendo que un espacio de memoria son 8 “espacios de estos” y que se conocen como bits, el sistema binario computacional empieza a tomar forma. En realidad el sistema es un estándar para poder hacer que toda la computación que se ejecuta en diferentes partes del mundo se entienda. De esta manera, un espacio de memoria de 8 bits, también llamado byte, es en sí, un número o un caracter. Este número en binario (colección de

1’s y 0’s) tiene su contraparte en el sistema decimal, por ende estos códigos tienen transformaciones entre ellos. Es curiosos que los caracteres (texto) también son números solo que la computadora, en lugar de interpretarlos en las operaciones como números, los integra como textos y las sumas, lejos de hacer la operación matemática, sólo coloca “uno al lado del otro”.

De esta forma, los números binarios tienen la siguiente correlación con los decimales:

Binario	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
	Etc.

Existe una manera de transformar cualquier número binario a decimal. El proceso es el siguiente:

1. Acomodar el número binario en N casillas, donde N es el número de 1’s y 0’s que tiene.
2. Abajo de cada 1 y 0, empezando de derecha a izquierda, corresponder 2 a la potencia que se identifica con el lugar en el que está.
3. Multiplicar cada 1 y 0 por la potencia de 2 y sumar todos los resultados.

La primer fila indica la posición del bit. Cuidado: empezar en **0**. La última fila es la suma de cada posición.

4	3	2	1	0
1	0	1	1	0
$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

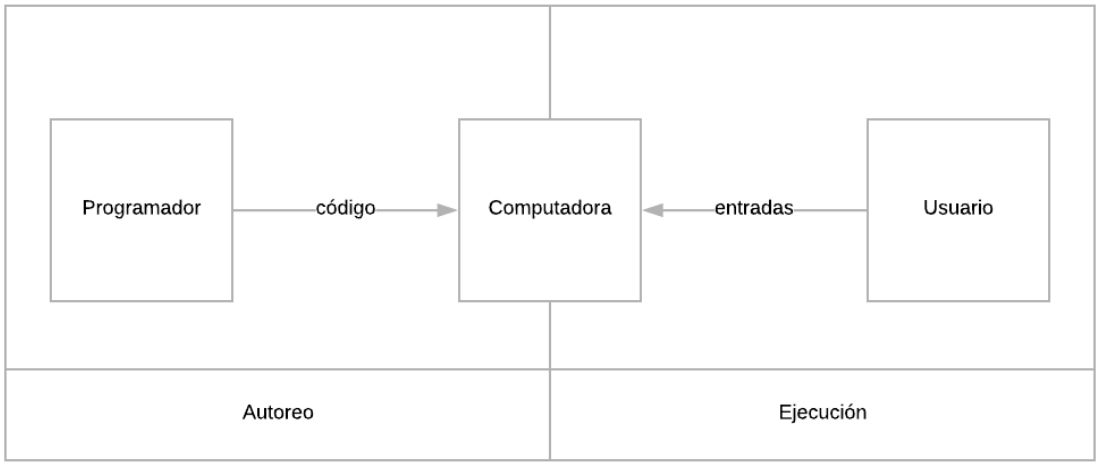
16*1 = 16	8* = 0	1*4 = 4	1*2= 2	0*1 = 0

El número 10110 binario es el 16+0+4+2+0 decimal. Es decir, es el 22.

Investigación: ¿como se podría sumar en binario? Y ¿restar?. También se podría dividir y multiplicar, pero los dos primeros son los más usados.

Hay otras 2 operaciones importantes con los números binarios, se llaman operaciones lógicas: AND y OR. ¿Cómo son?

Un elemento importante de cualquier programa es la habilidad para poder tomar respuestas del usuario y luego integrarlas en la programación. Para esto hay que ser claros en el modelo que una computadora propone:



En este sentido el programador coloca una serie de códigos que la computadora interpretará eventualmente como el programa. Por su cuenta, cuando la aplicación está “activa”, la persona que interactúa es un usuario. Generalmente el programador cambia entre estos dos roles continuamente tras cualquier cambio o adecuación.

Para probar esta retroalimentación se pueden usar códigos para interfase como lo son Mensajes o Alertas. Para mostrar la información, particularmente en programación gráfica, se utilizarán los dibujos y gráficas en pantalla.

Un primer acercamiento es el siguiente código:

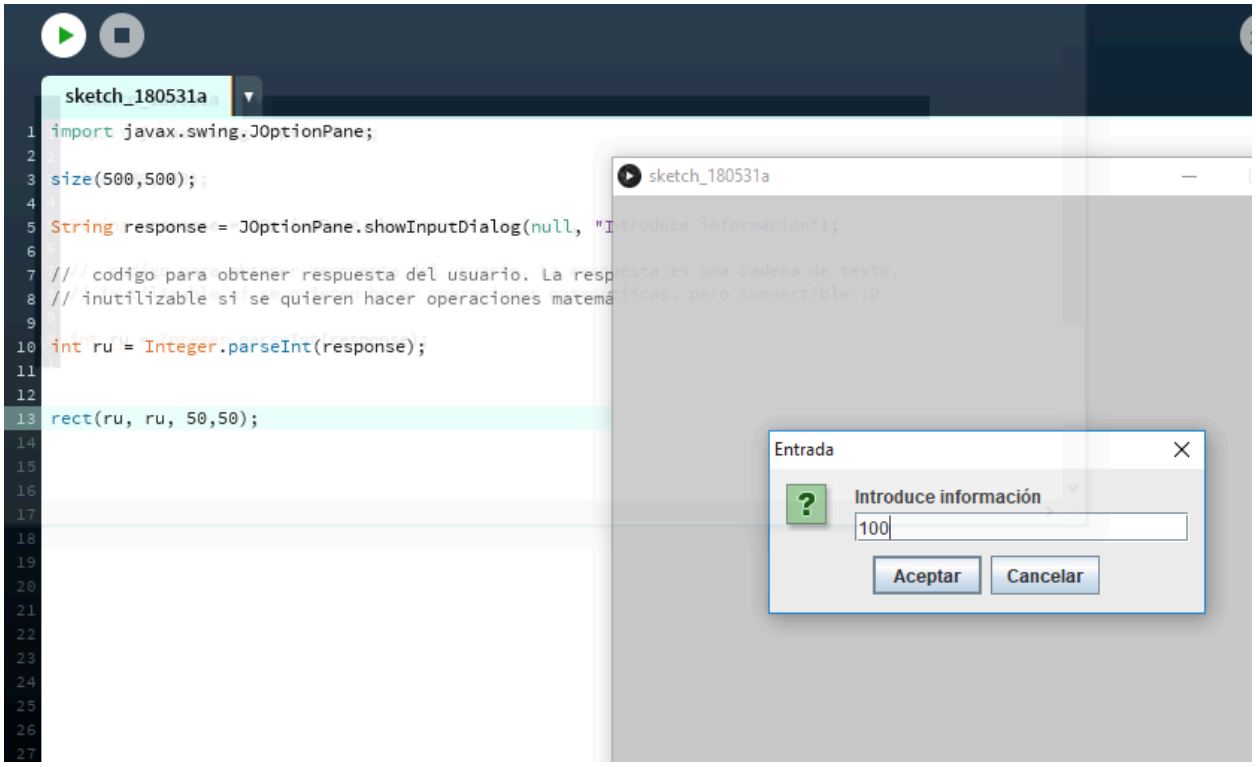
```
import javax.swing.JOptionPane;

size(500,500);

String response = JOptionPane.showInputDialog(null, "Introduce información");

//  codigo para obtener respuesta del usuario. La respuesta es una cadena de texto,
// inutilizable si se quieren hacer operaciones matemátiicas, pero convertible :D
```

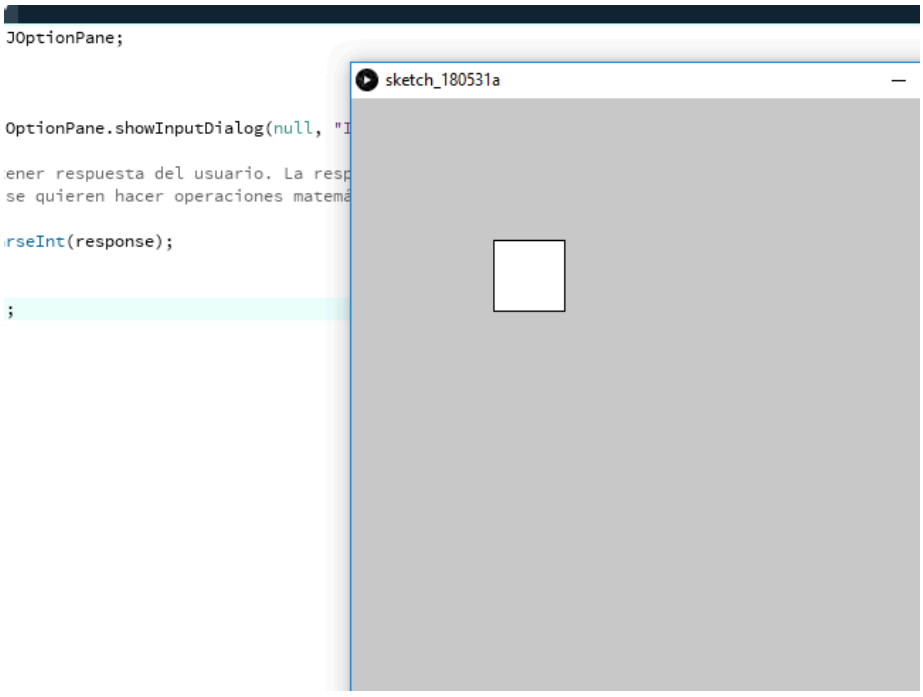
Generalmente el programador se coloca en el “lugar” de la computadora cuando escribe el código y “piensa” que él o ella es como tal la unidad de procesamiento. Este tipo de pensamiento permite que vaya “compilando” en tiempo real sus ideas en código. Es muy común oír a los programadores palabras como “obtengo la información”, “convierto los datos”, “rechazo en caso de error”, indicando no que ellos harán la operación sino la computadora.

Siguiendo con las opciones para poder “meter” información a la computadora, se podrán convertir los datos a partir de uno de herramientas de “parseo” que son convertidores en general. Para el caso de utilizar números a partir de la entrada del usuario se usará el convertidor “Integer.parseInt()”. Como en el siguiente código:

```
sketch_180531a
import javax.swing.JOptionPane;

size(500,500);

String response = JOptionPane.showInputDialog(null, "Introduce información");

//  codigo para obtener respuesta del usuario. La respuesta es una cadena de texto,
// inutilizable si se quieren hacer operaciones matemátiicas, pero convertible :D

int ru = Integer.parseInt(response);
```

Con lo cual, ya es posible usar entradas del usuario como elementos dentro de la programación.

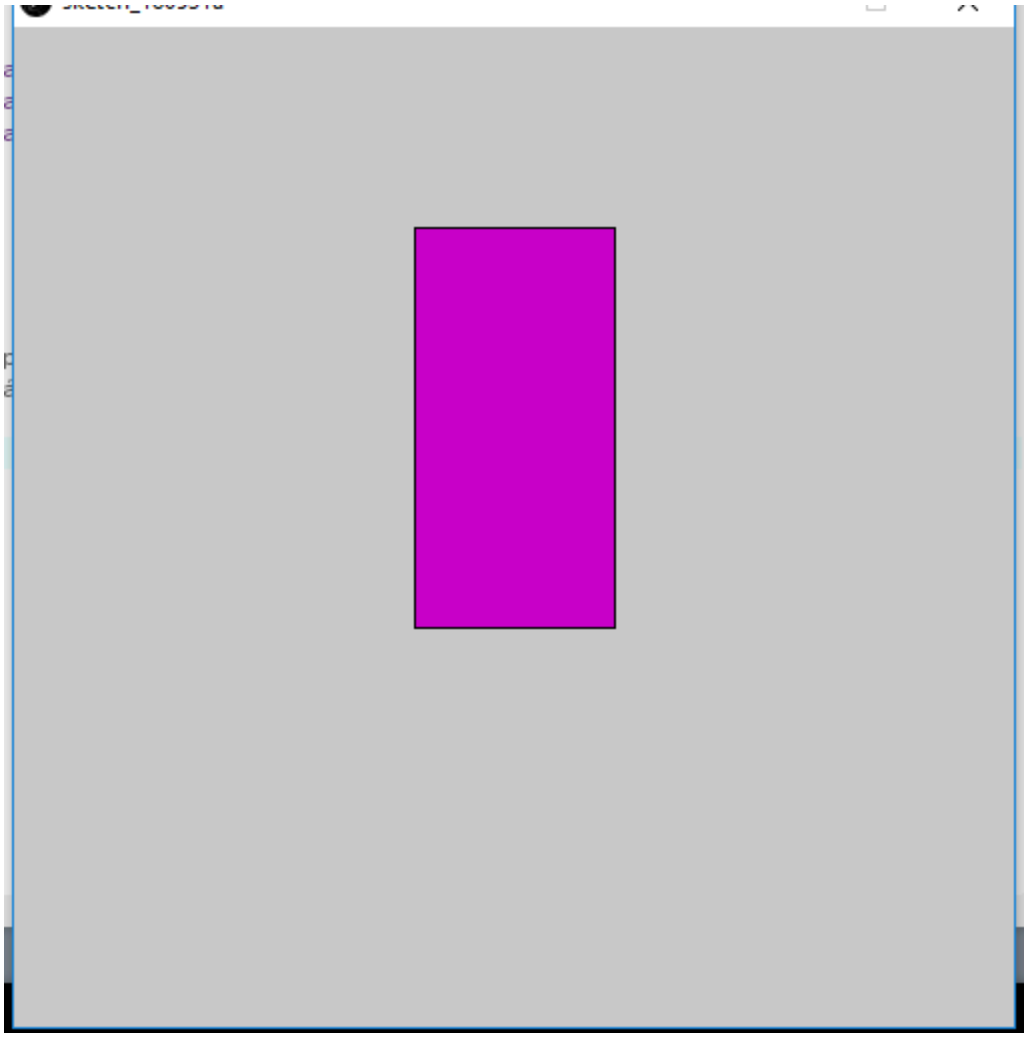




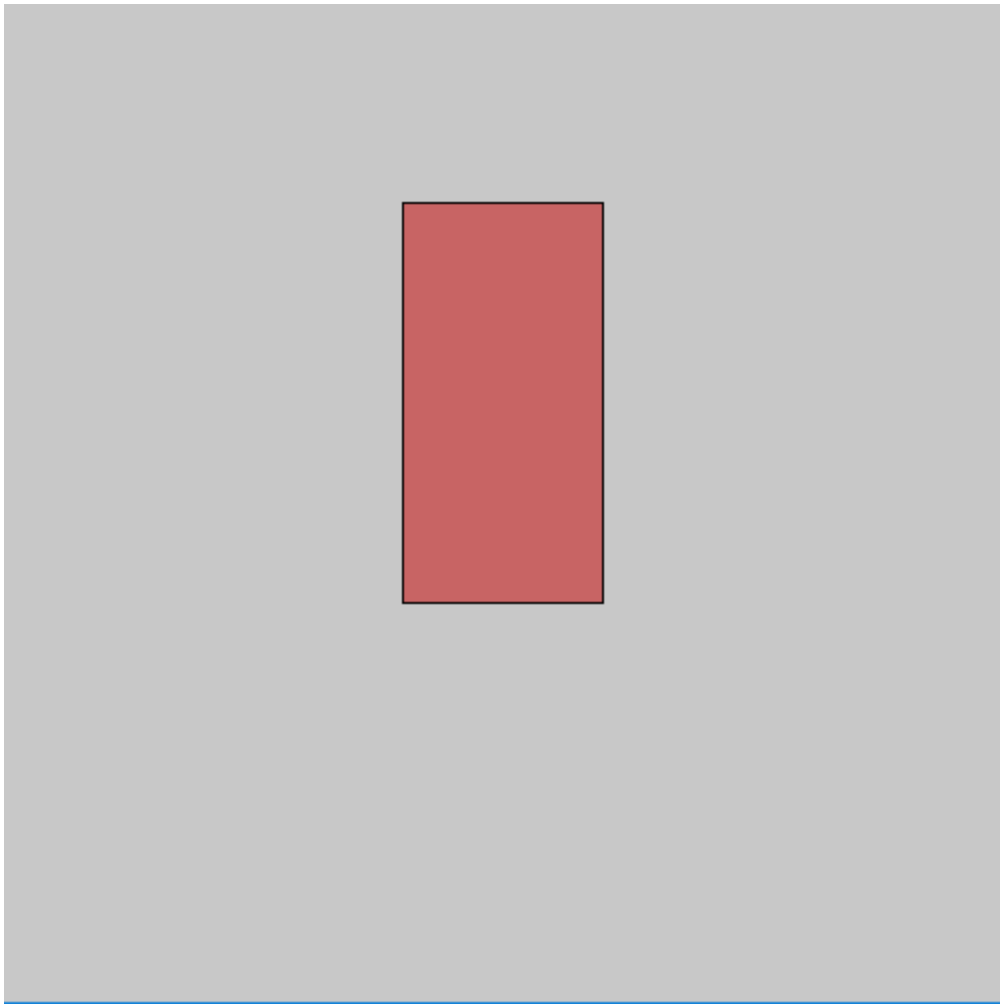
Ejercicio: modificar el ejercicio de las caras o los cuadros para preguntar al usuario en donde se quiere comenzar a pintar.

Problema:

Un brujo quiere hacer una poción usando tres sustancias, una roja, una verde y una azul. Y quiere hacerla con processing. Ayúdale haciendo un programa que le pida las cantidades de los tres teniendo como mínimo 0 ml y máximo 255 ml (de sustancia). Al final el programa tendría que mostrar la cantidad total de poción y el color resultante.



Ejemplo de poción 200 roja, 0 verde, 200 azul.



Ejemplo de poción 200,100,100.

Nota: las cantidades son directamente proporcionales al tamaño del rectángulo, sin embargo para que “quepa” en la pantalla puede ser necesario multiplicar el tamaño por 0.5 o 0.3.

De esta manera se logra interpretar el proceso completo de programación integrando variables, entradas, procesamiento y salida. En las unidades siguientes se verá la manera de hacer interactiva la aplicación y se introducirán conceptos como funciones y arreglos.

Nota matemática:

Las entradas que el usuario genera se toman como valores para variables que en matemáticas son valores que se toman por conocidos. Hay una diferencia importante entre las variables computacionales y matemáticas.

En la computadora una variable es un “nombre o alias” que puede tener un valor cambiable. En las matemáticas una variable es un conjunto posible de valores que pueden entrar dentro de una función o procedimiento. Estos conceptos son similares porque conceptualmente ambos proponen potencialmente todos los valores posibles, sin embargo en la computadora es necesario colocar los valores de las variables en una asignación directa, mientras que en las matemáticas es posible colocar asignaciones para posteriormente resolverlas como en los sistemas de ecuaciones.